THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of	
Inventor(s): David Hsing LIN	: Confirmation No. 5529
	:
U.S. Patent Application No. 10/823,845	: Group Art Unit: 2166
	:
Filed: April 14, 2004	: Examiner: Navneet K. Ahluwalia
For: METHOD AND APPARATUS FOR MULTI-PROCESS ACCESS TO A LINKED-LIST	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attn: BOARD OF PATENT APPEALS AND INTERFERENCES

BRIEF ON APPEAL

This brief is in furtherance of the Notice of Appeal, filed in this case on June 28, 2007.

The fees required under § 1.17(f) and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

TABLE OF CONTENTS

I. Real Party in Interest

The real party in interest is Hewlett-Packard Development Company, L.P., a Texas limited partnership.

II. Related Appeals and Interferences

There are no other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in this appeal.

III. Status of Claims

A. Total Number of Claims in Application

There is a total of 22 claims in the application, which are identified as claims 1-22.

B. Status of all the Claims

Claims 1-22 are pending.

Claims 1-22 are rejected.

C. Claims on Appeal

Claims on appeal are claims 1-22.

IV. Status of Amendments

There are no outstanding un-entered amendments before the Examiner.

V. Summary of Claimed Subject Matter

The present invention relates generally to a method and apparatus for multi-process access to a linked list.

Claim 1

Independent claim 1 recites a method for retrieving data comprising locking a linked list (Instant specification in at least paragraph 10 and FIG. 1, element 5). The method also comprises retrieving data from an element in the linked list and advancing to a subsequent element while a breakpoint is not encountered (Instant specification in at least paragraph 10 and FIG. 1, elements 10, 15, 20, and 25). The method also comprises marking the subsequent element in the linked-list as in-use after encountering a breakpoint (Instant specification in at least paragraphs 11-12, FIG. 1, element 15, and FIG. 2, element 30). The method also comprises creating a recommencement reference to the subsequent element; and unlocking the linked list (Instant specification in at least paragraphs 12-13 and FIG. 2, element 40).

Claim 5

Independent claim 5 recites a method for deleting an element from a linked list comprising updating a recommencement reference to the element to refer to a data element subsequent to the data element to be deleted for each element to be deleted which is determined to be in-use (Instant specification in at least paragraph 18 and FIG. 6, element 150). The method also comprises deleting the element (Instant specification in at least paragraph 18 and FIG. 6, element 155).

Claim 7

Independent claim 7 recites an apparatus for storing and retrieving data comprising a processor capable of executing an instruction sequence (Instant specification in at least paragraph 22, and FIG. 9, element 300). The apparatus also comprises a memory for storing an instruction sequence (Instant specification in at least paragraph 22, and FIG. 9, element 370). The apparatus also comprises an input unit for receiving data (Instant specification in at least paragraph 22, FIG. 9, element 310, and FIG. 10). The apparatus also comprises a first output unit for providing data according to a received data request (Instant specification in at least paragraph 22, FIG. 9, element 320, and FIG. 10, element 320) and one or more ancillary output units for providing data according to a received data request (Instant specification in at least paragraph 22, FIG. 9, element 350, FIG. 10, element 350).

The apparatus also comprises instruction sequences stored in the memory including: a data storage module (Instant specification in at least paragraphs 23-24 and FIGs. 9, 10, element 375) and a data service module (Instant specification in at least paragraphs 23, 25--30 and FIGs. 9, 10, element 380).

The data storage module, when executed by the processor, minimally causes the processor to: receive data from the input unit; allocate a data element to accommodate the data; create a reference to the data element; store the reference in at least one of a header pointer and a forward pointer included in a preceding data element; and store the data in the data element (Instant specification in at least paragraph 24).

The data service module, when executed by the processor, minimally causes the processor to: recognize a data request from the first output unit to the exclusion of all other data requests; provide data to the first output unit from a data element according to a data element reference and also advance the data element reference to a subsequent data element while a breakpoint is not encountered; mark a subsequent data element as in-use when a breakpoint is encountered; create a recommencement reference to a subsequent data element; and enable recognition of other data requests (Instant specification in at least paragraphs 25-28).

Claim 13

Independent claim 13 recites a computer readable medium (Instant specification in at least paragraph 31) having imparted thereon one or more instruction sequences for storing and retrieving data. The instruction sequences on the computer readable medium comprise a data storage module (Instant specification in at least paragraphs 23-24 and FIGs. 9, 10, element 375) that, when executed by a processor, minimally causes the processor to: receive data from an input unit; allocate a data element to accommodate the data; create a reference to the data element; store the reference in at least one of a header pointer and a forward pointer included in a preceding data element; and store the data in the data element (Instant specification in at least paragraph 24). The instruction sequences on the computer readable medium also comprise a data service module (Instant specification in at least paragraphs 23, 25--30 and FIGs. 9, 10, element 380) that, when executed by a processor, minimally causes the processor to: recognize a data request from a first output unit to the

exclusion of all other data requests; provide data to a first output unit from a data element according to a data element reference and also advance the data element reference to a subsequent data element while a breakpoint is not encountered; mark a subsequent data element as in-use when a breakpoint is encountered; create a recommencement reference to a subsequent data element; and enable recognition of other data requests (Instant specification in at least paragraphs 25-28).

Claim 19

Independent claim 19 recites an apparatus for storing and retrieving data comprising means for locking a linked list (Instant specification in at least paragraphs 23, 25, data service module 380 instructions stored in memory 370 for execution by processor 300, and FIGs. 9, 10, elements 375, 370, and 300). The apparatus also comprises means for retrieving data from an element in the linked list and also advancing to a subsequent element while a breakpoint is not encountered (Instant specification in at least paragraphs 23 and 26, data service module 380 instructions stored in memory 370 for execution by processor 300, and FIGs. 9, 10, elements 375, 370, and 300). The apparatus also comprises means for marking the subsequent element in the linked-list as in-use when a breakpoint is encountered (Instant specification in at least paragraphs 23, 27, data service module 380 instructions stored in memory 370 for execution by processor 300, and FIGs. 9, 10, elements 375, 370, and 300). The apparatus also comprises means for creating a recommencement reference to the subsequent element (Instant specification in at least paragraphs 23, 27, data service module 380 instructions stored in memory 370 for execution by

processor 300, and FIGs. 9, 10, elements 375, 370, and 300). The apparatus also comprises means for unlocking the linked list (Instant specification in at least paragraphs 23, 27, data service module 380 instructions stored in memory 370 for execution by processor 300, and FIGs. 9, 10, elements 375, 370, and 300).

VI. Grounds of Rejection to be Reviewed on Appeal

A. The issue is whether claims 1-22 are unpatentable under 35 U.S.C 102(e) as being unpatentable over *Gao et al*. (US 6,898,650).

VII. Argument

A. Was the PTO correct in rejecting claims 1-22 under 35 U.S.C. 102(e) as being unpatentable over *Gao*?

Claim 1

The rejection of claims 1-22 as being unpatentable over *Gao* is believed to be incorrect and is hereby traversed. A rejection based on 35 U.S.C. §102 requires every element of the claim to be included in the reference, either directly or inherently. Because *Gao* fails to disclose every element of the claim, reversal of the rejection is respectfully requested.

The Patent and Trademark Office (PTO) asserts that claims 1-22 are anticipated under 35 USC 102(e) by *Gao et al.* (US 6,898,650). Final Official Action (FOA) mailed May 15, 2007 at pages 2 and 3. There are at least two reasons claim 1 is patentable over *Gao*.

First, *Gao* fails to disclose or suggest "marking the subsequent element in the linked-list as in-use after encountering a breakpoint" as recited in claim 1. The PTO asserts that *Gao*, at column 3, lines 39-50, discloses the claimed limitation. This is incorrect.

The PTO-identified portion of *Gao*, reproduced herein for ease of reference, states:

> FIG. 3 shows a container for a queue according to the preferred embodiment of the invention implemented on the computer system of FIG. 1. In FIG. 3, container 305 includes in-use and data valid flags 310 and 315, data field 320, and next pointer 325. In-use flag 310 indicates whether the container is being used at the current time (in other words, whether or not the container is available for use). In-use flag 310 is the preferred embodiment for a container lock, which allows only one client to use a container at a time. Data valid flag 315 indicates whether a

> container holds valid data. But data valid flag 315 is not absolutely required, and can be part of data field 320.

The above portion of *Gao* appears to describe the use of in-use flag 310 and data valid flag 315 to indicate whether a container is being used and whether a container holds valid data, respectively. However, this does not meet the claimed limitation of the present claimed subject matter. The above portion of *Gao* fails to disclose "encountering a breakpoint" and therefore also fails to disclose marking a subsequent element in a list as in-use after encountering a breakpoint. For at least this reason, reversal of the rejection is respectfully requested.

Further, according to *Gao* at column 4, lines 36-40, a client attempts to set in-use flag 310 using an atomic set and swap operation "to try to lock the container," "so that no other client can use the container." Thus, the in-use flag of *Gao* does not appear to be marked after encountering a breakpoint. For at least this reason, reversal of the rejection is respectfully requested.

Second, *Gao* fails to disclose or suggest "creating a recommencement reference to [a] subsequent element" as recited in claim 1. As described in the instant specification at page 6, paragraph 12, creation of a recommencement reference allows a first process to "unlock the linked-list (step 45), thereby allowing an opportunity for a second process to gain control over the linked-list." After the first process regains control over the linked-list, the first process is able to "determine[] a subsequent element in the linked-list according to the recommencement reference that points to a subsequent element (step 55)." Instant specification at page 7, paragraph 13.

The PTO asserts that *Gao*, at column 2, lines 46-58, column 3, lines 9-20 and lines 51-59, and column 4, lines 36-49, discloses the claimed limitation. FOA at page 2, fourth full paragraph and page 3, third full paragraph. This is incorrect.

Column 2, lines 46-58 of *Gao*, reproduced herein for ease of reference, states:

> FIG. 2 shows a queue head for a queue according to the preferred embodiment of the invention implemented on the computer system of FIG. 1. In FIG. 2, queue head 205 includes read/write lock 210

> (sometimes also called a queue lock), counter 215, and next pointer 220. Read/write lock 210 indicates whether the queue is locked for reading or writing. When clients wish to access containers in the queue, they lock the queue for reading. There can be as many simultaneous clients reading the queue as desired.

The above portion of *Gao* appears to describe the use of a queue head 205 which appears to be an object pointing to the beginning (or head) of the queue, e.g., queue 405 of FIG. 4. There appears to be no disclosure of the next pointer as a recommencement reference to a subsequent element in the linked-list as claimed.

Nor does there appear to be any disclosure of creating a recommencement reference to a subsequent element through the use of the next pointer in *Gao*. That is, the next pointer appears to be set to point to the head of the queue and not a subsequent element in the queue.

Column 3, lines 9-20 and 51-59 of *Gao*, reproduced herein for ease of reference, state:

> Returning to FIG. 2, next pointer 220 points to one of the containers in the queue. Note that it does not matter which container in the queue next pointer 220 points to, so long as all containers are accessible. Thus, the organization of the queue is not relevant to the invention, and the invention is equally applicable to different queue implementations. For example, the queue can be structured as a singly linked list, a doubly linked list, a circular list, or an array. Further, the invention is applicable to priority queues (queues in which the containers are each assigned a priority, and containers with higher priorities are used before containers with lower priorities.
>
> . . .
>
> Returning to FIG. 3, data field 320 stores the data in the container. The type of data stored in the container is generally not limited, although queues designed to store specific types of data are possible. Similarly, the amount of data stored in the container is generally not limited. Finally next pointer 325 points to the next container in the queue (or, if the queue has an end and the current container is the last container in the queue, next pointer 325 is a null pointer).

The above portions of *Gao* appear to describe the use of next pointer 220 to point to the queue without describing the creation of the next pointer as a reference to a subsequent element. The second of the above portions, i.e., column 3, lines 51-59, appear to be directed to describing the use of the next pointer 325 of container 305 and not queue head 205. That is, the PTO appears to be confusing two separate discussions in the reference. This relied-upon portion is inapplicable with respect to the next pointer 220 identified by the PTO. For at least this reason, reversal of the rejection is respectfully requested.

Further, neither of next pointer 220 or 325 may be used as a recommencement reference as described according to the instant specification, i.e., after a first process regains control over the linked-list from a second process, the first process is able to "determine[] a subsequent element in the linked-list according to the recommencement reference that points to a subsequent element (step 55)." Instant specification at page 7, paragraph 13.

Column 4, lines 36-49, reproduced herein for ease of reference and convenience, state as follows:

> At step 510, the client locates a container in the queue. At step 515, attempts to lock the container, so that no other client can use the container. In the preferred embodiment, an atomic set and swap operation is used to try to lock the container by setting the in-use flag to 1. An atomic set and swap operation sets a field to the given value and returns the old value atomically (in one indivisible computer operation). Many modern computer systems (e.g., IBM mainframes) have such instructions, and most MP environments (e.g., Novell's Multiple Processor Kernel) include such functions. Generally, the atomic set and swap operation will return the value of the field being accessed to the caller; the value returned gives the caller an indication of whether the operation succeeded.

The PTO-identified portion of *Gao* appears to describe using an atomic set and swap operation to attempt to lock access to a container in a queue. There appears to be no disclosure of creation of a recommencement reference to a subsequent element in the linked-list as claimed.

Further, *Gao* fails to specify how the "client locates a container in the queue" and instead, based on the listing of Table 14 appears to describe traversal of the queue by following the next pointer 325 of each container 305 instead of use of queue head next pointer 220 as asserted by the PTO. For at least this reason, reversal of the rejection is respectfully requested.

Based on at least each of the foregoing reasons, claim 1 is patentable over *Gao* and the rejection is respectfully requested to be reversed. Claims 2-4 depend, either directly or indirectly, from claim 1, include further limitations, and are patentable over *Gao* for at least the reasons advanced above with respect to claim 1. The rejection of claims 2-4 should be reversed.

Claim 5

Gao fails to disclose or suggest at least "updating a recommencement reference" as claimed in claim 5.

As set forth above with respect to claim 1, *Gao* appears to describe the use of an in-use flag, however, *Gao* fails to disclose or suggest a recommencement reference as claimed in claim 5. The *Gao* in-use flag is not used to recommence traversal of a linked-list. For at least this reason, reversal of the rejection is respectfully requested.

Claim 6 depends, either directly or indirectly, from claim 5, includes further limitations, and is patentable over *Gao* for at least the reasons advanced above with respect to claim 5. The rejection of claim 6 should be reversed.

Claims 7 and 13

Gao fails to disclose or suggest at least "create a recommencement reference to a subsequent data element" as claimed in claim 7 and 13.

Claims 7 and 13 are patentable over *Gao* for at least reasons similar to those advance above with respect to claim 1 and the rejection of claims 7 and 13 should be reversed.

Claims 8-12 and 14-18 depend, either directly or indirectly, from claims 7 and 13, respectively, include further limitations, and are patentable over *Gao* for at least the reasons advanced above with respect to claims 7 and 13. The rejection of claims 8-12 and 14-18 should be reversed.

Claim 19

Gao fails to disclose or suggest at least "means for creating a recommencement reference to the subsequent element" as claimed in claim 19. Similar to the above reasons presented with respect to claim 1, claim 19 is patentable over *Gao* and the rejection should be reversed.

Claims 20-22 depend, either directly or indirectly, from claim 19, include further limitations, and are patentable over *Gao* for at least the reasons advanced above with respect to claim 19. The rejection of claims 20-22 should be reversed.

VIII. Conclusion

Each of the PTO's rejections has been traversed. Appellant respectfully submits that all claims on appeal are considered patentable over the applied art of record. Accordingly, reversal of the PTO's Final Rejection is believed appropriate and courteously solicited.

If for any reason this Appeal Brief is found to be incomplete, or if at any time it appears that a telephone conference with counsel would help advance prosecution, please telephone the undersigned, Appellant's attorney of record.

To the extent necessary, a petition for an extension of time under 37 C.F.R. 1.136 is hereby made. Please charge any shortage in fees due in connection with the filing of this paper, including extension of time fees, to Deposit Account 08-2025 and please credit any excess fees to such deposit account.

Reversal of the rejection is in order.

Respectfully submitted,
David Hsing LIN

By: 
Randy A. Noranbrock
Registration No. 42,940
Telephone: 703-684-1111

HEWLETT-PACKARD COMPANY
IP Administration
Legal Department, M/S 35
P.O. Box 272400
Fort Collins, CO 80528-9599
Telephone: 970-898-7057
Facsimile: 281-926-7212
Date: **August 28, 2007**
RAN/

IX. Claims Appendix

Listing of Claims:

1. A method for retrieving data comprising:

locking a linked list;

retrieving data from an element in the linked list and advancing to a subsequent element while a breakpoint is not encountered;

marking the subsequent element in the linked-list as in-use after encountering a breakpoint;

creating a recommencement reference to the subsequent element; and

unlocking the linked list.

2. The method of claim 1 further comprising:

locking the linked list;

determining a subsequent element in the linked list according to the recommencement reference; and

retrieving data from the determined subsequent element.

3. The method of claim 1 wherein creating a recommencement reference to the subsequent element comprises:

retrieving a pointer to the subsequent element;

determining a process identifier for a current process; and

associating the pointer with the process identifier.

4. The method of claim 1 wherein marking the subsequent element in the linked-list as in-use comprises maintaining a count of the quantity of processes that require additional access to the element.

5. A method for deleting an element from a linked list comprising:

updating a recommencement reference to the element to refer to a data element subsequent to the data element to be deleted for each element to be deleted which is determined to be in-use; and

deleting the element.

6. The method of claim 5 wherein updating a recommencement reference to the element comprises:

discovering a pointer associated with a process identifier;

disassociating the process identifier from the pointer;

determining a pointer to a subsequent element; and

associating the process identifier with the newly determined pointer.

7. An apparatus for storing and retrieving data comprising:

processor capable of executing an instruction sequence;

memory for storing an instruction sequence;

input unit for receiving data;

first output unit for providing data according to a received data request;

one or more ancillary output units for providing data according to a received data request;

instruction sequences stored in the memory including:

data storage module that, when executed by the processor, minimally causes the processor to:

receive data from the input unit;

allocate a data element to accommodate the data;

create a reference to the data element;

store the reference in at least one of a header pointer and a forward pointer included in a preceding data element;

and store the data in the data element;

data service module that, when executed by the processor, minimally causes the processor to:

recognize a data request from the first output unit to the exclusion of all other data requests;

provide data to the first output unit from a data element according to a data element reference and also advance the data element reference to a subsequent data element while a breakpoint is not encountered;

mark a subsequent data element as in-use when a breakpoint is encountered;

create a recommencement reference to a subsequent data element; and

enable recognition of other data requests.

8. The apparatus of claim 7 wherein the data service module, when executed by the processor, further minimally causes the processor to:

recognize a data request from the first output unit to the exclusion of all other data requests; and

provide data to the first output unit from a data element according to the recommencement reference.

9. The apparatus of claim 7 wherein the data service module causes the processor to create a recommencement reference by minimally causing the processor to:

retrieve a pointer to a data element subsequent to a current data element;

determine an identifier associated with the data request received from the first output unit; and

store the retrieved pointer and the determined identifier in an associative manner.

10. The apparatus of claim 7 wherein the data service module causes the processor to mark a subsequent data element as in-use by minimally causing the processor to increment a use counter included in a subsequent data element.

11. The apparatus of claim 7 wherein the data service module further minimally causes the processor to receive a delete data request from an output unit by minimally causing the processor to:

determine if a data element to be deleted is in-use;

update a recommencement reference to refer to a data element that is subsequent to the data element to be deleted; and

delete the data element according to the received delete data request.

12. The apparatus of claim 11 wherein the data service module causes the processor to update a recommencement reference by minimally causing the processor to:

discover a pointer according to a data request identifier; and

replace the pointer with a pointer to a data element that is subsequent to the data element to be deleted.

13. A computer readable medium having imparted thereon one or more instruction sequences for storing and retrieving data comprising:

data storage module that, when executed by a processor, minimally causes the processor to:

receive data from an input unit; allocate a data element to accommodate the data;

create a reference to the data element;

store the reference in at least one of a header pointer and a forward pointer included in a preceding data element; and

store the data in the data element;

data service module that, when executed by a processor, minimally causes the processor to:

recognize a data request from a first output unit to the exclusion of all other data requests;

provide data to a first output unit from a data element according to a data element reference and also advance the data element reference to a subsequent data element while a breakpoint is not encountered;

mark a subsequent data element as in-use when a breakpoint is encountered;

create a recommencement reference to a subsequent data element; and

enable recognition of other data requests.

14. The computer readable medium of claim 13 wherein the data service module, when executed by a processor, further minimally causes the processor to:

recognize a data request from a first output unit to the exclusion of all other data requests; and

provide data to a first output unit from a data element according to the recommencement reference.

15. The computer readable medium of claim 13 wherein the data service module causes a processor to create a recommencement reference by minimally causing the processor to:

retrieve a pointer to a data element subsequent to a current data element;

determine an identifier associated with a data request received from a first output unit; and

store the retrieved pointer and the determined identifier in an associative manner.

16. The computer readable medium of claim 13 wherein the data service module causes a processor to mark a subsequent data element as in-use by minimally causing the processor to increment a use counter included in a subsequent data element.

17. The computer readable medium of claim 13 wherein the data service module further minimally causes the processor to receive a delete data request from an output unit by minimally causing the processor to:

determine if a data element to be deleted is in-use;

update a recommencement reference to refer to a data element that is subsequent to the data element to be deleted; and

delete the data element according to the received delete data request.

18. The computer readable medium of claim 17 wherein the data service module causes the processor to update a recommencement reference by minimally causing the processor to:

discover a pointer according to a data request identifier; and

replace the pointer with a pointer to a data element that is subsequent to the data element to be deleted.

19. An apparatus for storing and retrieving data comprising:

means for locking a linked list;

means for retrieving data from an element in the linked list and also advancing to a subsequent element while a breakpoint is not encountered;

means for marking the subsequent element in the linked-list as in-use when a breakpoint is encountered;

means for creating a recommencement reference to the subsequent element; and

means for unlocking the linked list.

20. The apparatus of claim 19 further comprising:

means for locking the linked list;

means for determining a subsequent element in the linked list according to the recommencement reference; and

means for retrieving data from the determined subsequent element.

21. The apparatus of claim 19 further comprising a means for deleting an element in the linked-list.

22. The apparatus of claim 21 wherein the means for deleting an element comprises:

means for determining if the element to be deleted is in-use;

means for updating a reference to the element to refer to a subsequent element in the linked list when the element is in-use; and

means for deleting the element.

X. Evidence Appendix

None.

XI. Related Proceedings Appendix

None.